Switching from Python based 3D viewer to Javascript based 3D viewer for shape Visualization

1. Background & Initial Approach

The original objective was to build a 3D model viewer that could render, measure, and display 3D models from .stl files interactively, with measurements shown upon user interaction. The initial approach used Python-based libraries, particularly with PyVista and Trimesh, to handle 3D rendering and geometric analysis tasks.

2. Challenges with Python-Based Libraries

Several issues emerged with the Python approach:

- Interactivity and Visualization Complexity: Python libraries like PyVista and Trimesh focus on scientific analysis and geometry processing. However, they offer limited capabilities for building interactive, browser-based applications. Additionally, managing real-time interactivity (such as hovering effects, dynamic annotations, and measurements) proved challenging with these libraries.

- Performance and Compatibility: Web-based rendering in Python often requires tools like Plotly, which integrate Python with JavaScript for interactivity but tend to be slower and less interactive compared to native JavaScript libraries designed specifically for the web.

- Rendering in the Browser: Python-based solutions generally require server-side processing for 3D rendering and interactivity, which introduces latency and complex server-client communication. Direct rendering of 3D models in the browser with Python is less straightforward than JavaScript.

Due to these limitations, we opted to switch to a JavaScript-based approach to leverage its native compatibility with browser rendering and interactivity.

3. Solution with JavaScript and Three.js

Using JavaScript, specifically with Three.js and React Three Fiber libraries, we successfully implemented a 3D viewer with robust interactivity and efficient rendering.

- JavaScript-Based Framework for 3D Rendering: The new approach uses Three.js, a lightweight JavaScript library optimized for 3D rendering in the browser. This framework provides extensive support for handling .stl files and rendering them directly in the browser, with built-in support for interactivity.

- React Three Fiber and Drei: By using React Three Fiber (a React renderer for Three.js), we could easily structure our components and manage state in a React-friendly manner, improving

maintainability and scalability. @react-three/drei provided convenient components like OrbitControls and Html for creating complex interactions and overlays without needing to write custom code.

- Interactive Measurements and Hover Effects: With Three.js and React Three Fiber, we could create the desired interactivity, such as displaying measurements on hover. The viewer measures bounding box dimensions using STLLoader for loading geometry and calculates dimensions based on bounding boxes, making it efficient and responsive.

4. How the New Solution Works

The new solution comprises the following components:

1. **Model Upload and Parsing**: Users upload .stl files, which are converted to URLs and passed to Three.js's STLLoader for parsing. STLLoader loads the geometry directly in the browser, enabling immediate rendering.

2. **Interactive 3D Model Rendering**: Using Canvas from React Three Fiber, we set up a 3D scene with ambient and point lights, a camera, and controls to allow users to explore the model in 3D. OrbitControls enables intuitive navigation around the model, enhancing the user experience.

3. **Hover-Based Measurement Display**: The Model component listens for pointer events. When the user hovers over the model, it calculates dimensions using bounding boxes, then displays measurements as a floating HTML overlay at the pointer location. This dynamic response is fast and smooth, providing a seamless interactive experience.

4. **Sidebar and Measurement Display**: The Sidebar component allows users to upload files and displays the measurements of the selected model. Measurements are updated in real time based on user interactions, providing an intuitive interface.

Conclusion

Switching to JavaScript with Three.js and React Three Fiber addressed the limitations of Python for web-based interactivity and enabled the following benefits:

- Efficient Rendering and Real-Time Interactivity: The model renders directly in the browser, and updates are immediate, with minimal latency.
- Enhanced User Experience: Users can intuitively interact with the model, navigate the 3D space, and view measurements dynamically, creating an engaging and functional tool.
- Simplified Development and Maintenance: Leveraging JavaScript-based libraries within React makes it easier to maintain, extend, and integrate additional features in the future.

This approach ensures that the 3D viewer meets the original goals and performs optimally in a browser environment.